

# 《Logicalization of BERT》

YKY

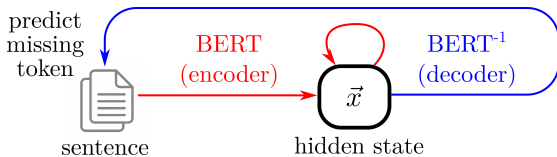
August 1, 2020

# Table of contents

- 1 BERT 的革命性意义：「闭环路训练」
- 2 BERT 的内部结构
- 3 Symmetry in logic
- 4 Symmetric neural network
- 5 BERT 的逻辑化
- 6 Attention 是什么？
- 7 谓词 (predicates) 与 命题 (propositions)
- 8 Attention 在命题之间的作用
- 9 “Attention is all you need” ?
- 10 应用：content-addressable long-term memory
- 11 应用：知识图谱 (knowledge graphs)

# BERT 的革命性意义：「闭环路训练」

- BERT 利用平常的文本 induce 出知识，而这 representation 具有 通用性 (universality)：



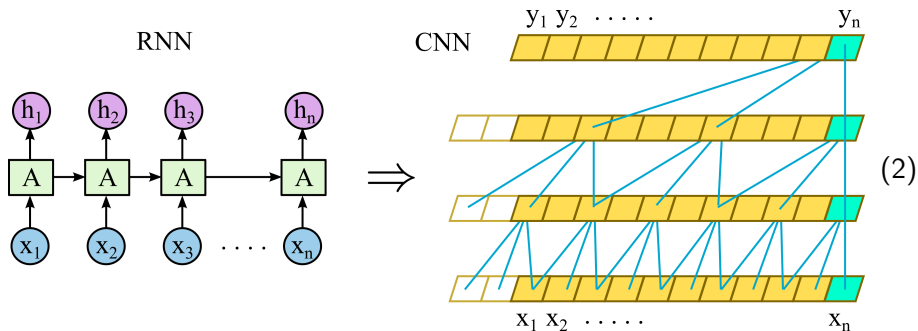
换句话说：隐状态的 representation 压缩了句子的意思，而它可以应用在别的场景下

- This implies that human-level AI can be *induced* from existing corpora, 而不需重复 像人类婴儿成长的学习阶段
- 这种训练方法是较早的另一篇论文提出，它并不属于 BERT 的内部结构

# BERT 的内部结构

其实，BERT 也是混合了很多技巧 发展而成的：

- BERT 基本上是一个 seq-to-seq 的运算过程
- Seq-to-seq 问题最初是用 RNN 解决的
- 但 RNN 速度较慢，有人提出用 CNN 取代：



- CNN 加上 attention mechanism 变成 Transformer
- 我的目标是重复这个思路，但引入 symmetric NN 的结构

# Symmetry in logic

- 词语 组成 句子, 类比於 逻辑中, 概念 组成 逻辑命题
- 抽象地说, 逻辑语言 可以看成是一种有 2 个运算的 代数结构, 可以看成是 加法  $\wedge$  和 乘法  $\cdot$ , 其中 乘法 是不可交换的, 但加法 可交换

- 例如:

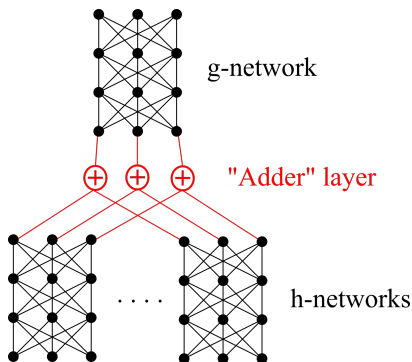
$$\begin{aligned} A \wedge B &\equiv B \wedge A \\ \text{下雨} \wedge \text{失恋} &\equiv \text{失恋} \wedge \text{下雨} \end{aligned} \tag{3}$$

- Word2Vec 也是革命性的; 由 Word2Vec 演变成 Sentence2Vec 则比较容易, 基本上只是 向量的 合并 (concatenation); Sentence 对应於 逻辑命题
- 但 命题的 集合 需要用 symmetric NN 处理, 因为 集合的元素 是顺序无关的

# Symmetric neural network

- Symmetric NN 问题 已经由 两篇论文解决了:  
[PointNet 2017] [DeepSets 2017]
- Any symmetric function can be represented by the following form (a special case of the Kolmogorov-Arnold representation of functions):

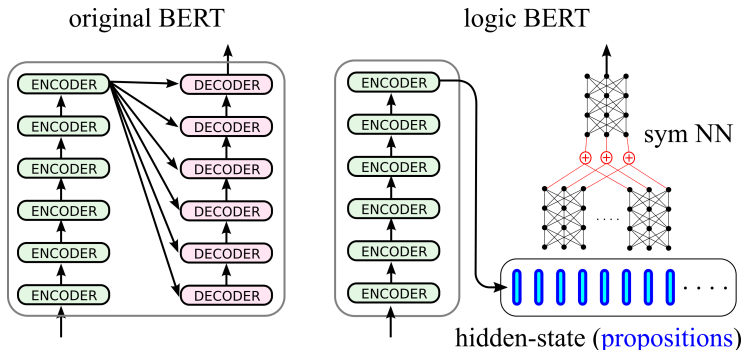
$$f(x, y, \dots) = g(h(x) + h(y) + \dots) \quad (4)$$




(5)

# BERT 的逻辑化

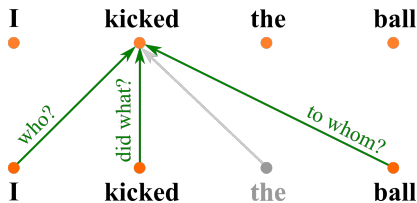
- 可以将 BERT 的 隐状态 变成 “set of propositions” 的形式，方法是将 原来的 decoder 变成 sym NN:



- 这时，输出对  的交换不变性 is automatically satisfied by the architecture of the decoder
- 旧的 encoder 可以照旧使用，，因为它是一个 universal seq-2-seq mapping
- 因为后半部改变了，error propagation 会令 representation 也改变
- 当然，这个想法有待实验证实 😊

# Attention 是什么?

- 注意力 最初起源於 Seq2seq, 后来 BERT 引入 self-attention
- Attention 的本质就是 **加权**, 权值 可以反映 模型 **关注** 的点
- For each input, the attention mechanism weighs the **relevance** of every other input and draws information from them accordingly to produce the output
- 在 BERT 里, attention 是一种 **words** 之间的关系:



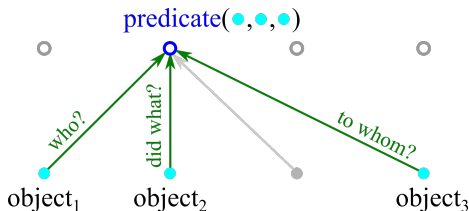
(7)

- 但, 从逻辑的角度看, word  $\neq$  命题
- 在逻辑学上, 必需分清 命题**内部** 与 命题**之间** 这两个层次, 非常关键!



# 谓词 (predicates) 与 命题 (propositions)

- “Predicate” 来自拉丁文「断言」的意思
- 逻辑里, predicate 代表一个 没有主体 / 客体 的断言, 换句话说, 是一个有「洞」的命题
- 命题** = **谓词** (predicate) + **主体 / 客体** (统称 objects)
- 例如: Human(John), Loves(John, Mary)
- 从逻辑的角度看, attention 的输出可以看成是 predicate 和 objects 的**结合**:



(8)

- 形象地说:

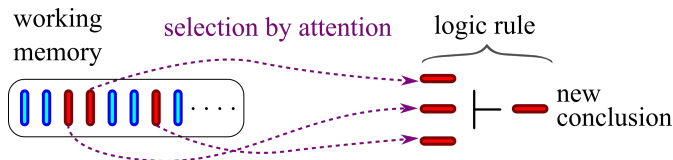
predicate + objects = proposition

$\bigcirc + \bullet \bullet \bullet \dots = \text{—}$

(9)

# Attention 在命题之间的作用

- 类似地，**高层的** attention 可以处理 **命题之间** 的关系，但这时 attention 机制似乎有严重的不足之处
- 我们希望 attention 做到的是 **选择** 有关联的命题，去做逻辑推导：



- 但，逻辑上有关联的东西 未必是相似的
- 例如：尿急  $\wedge$  不在厕所  $\Rightarrow$  忍著；但「尿急」和「不在厕所」并没有 *a priori* 的关系

# “Attention is all you need” ?

- At the propositional level, attention 要用 weight matrix 记住各种命题之间的相关性, 这种做法似乎 缺乏效率
- 例如, 假设  $q$  是被关注的逻辑命题,  $p_1, p_2, \dots$  是一些可能相关的命题,  $\bowtie$  表示 matching by attention, 我们希望输出一些 cases:

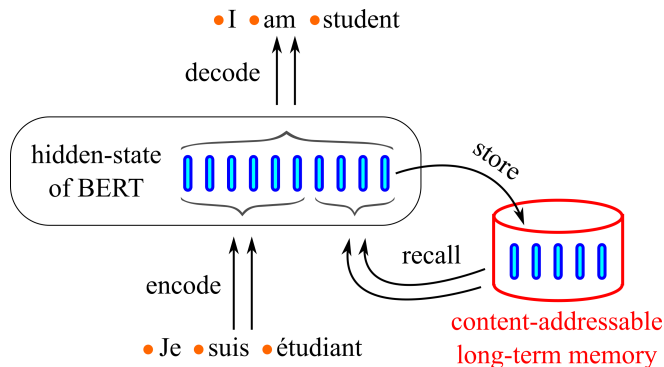
$$q \bowtie p_1, p_2, \dots \mapsto \text{case} \quad (11)$$

但每个 case 需要 矩阵的一行 储存

- 这是一种 “flat” representation of cases
- 而如果我们企图使用 hierarchical 的方法处理, 这个方向 会越来越变得像 deep learning, 还不如干脆使用 神经网络!
- 换句话说: 直接使用 deep symmetric NN, 在 NN 内部 学习如何 选择 相关的命题

## 应用：content-addressable long-term memory

- 以前 BERT 的隐状态 没有逻辑结构，我们不是很清楚它的内容是什么；逻辑化之后，BERT 内部的命题可以储存在 **长期记忆** 中：



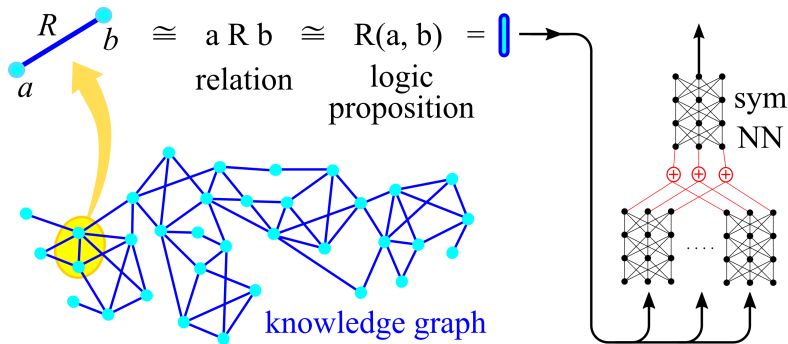
(12)

例如：「太阳是热的」、「水向下流」是经常正确的命题

- 这种系统 已非常接近 strong AI，而这是有赖 **逻辑化** 才能做到的
- Content-addressable memory 的想法来自 Alex Graves *et al* 的 Neural Turing Machine [2014]

## 应用：知识图谱 (knowledge graphs)

- 知识图谱 不能直接输入神经网络，它必需分拆成很多 edges，每个 edge 是一个 **关系**，也是一个 **逻辑命题**；也可以说 “graphs are isomorphic to logic”



- 而这些 edges 似乎必需用 **symmetric** NN 处理，因为它们是 permutation invariant

# References

欢迎提问和讨论 😊

- [1] Alex Graves, Greg Wayne, and Ivo Danihelka. “Neural Turing Machines”. In: *CoRR* abs/1410.5401 (2014). arXiv: 1410.5401. URL: <http://arxiv.org/abs/1410.5401>.
- [2] Qi et al. “Pointnet: Deep Learning on Point Sets for 3D Classification and Segmentation”. In: *CVPR* (2017). <https://arxiv.org/abs/1612.00593>.
- [3] Zaheer et al. “Deep sets”. In: *Advances in Neural Information Processing Systems* 30 (2017), pp. 3391–3401.